

# Machine Learning Models with Fault Tree Analysis for Explainable Failure Detection in Cloud Computing

Rudolf Hoffmann<sup>a</sup> and Christoph Reich<sup>b</sup>

*Institute for Data Science, Cloud Computing and IT Security, Furtwangen University, Germany*

**Keywords:** Cloud Computing, Reliability, Machine Learning, AI, XAI, Transparency, Explainability, Surrogate Model, Failure Detection, Fault Tree Analysis, Root Cause Analysis.

**Abstract:** Cloud computing infrastructures availability rely on many components, like software, hardware, cloud management system (CMS), security, environmental, and human operation, etc. If something goes wrong the root cause analysis (RCA) is often complex. This paper explores the integration of Machine Learning (ML) with Fault Tree Analysis (FTA) to enhance explainable failure detection in cloud computing systems. We introduce a framework employing ML for FT selection and generation, and for predicting Basic Events (BEs) to enhance the explainability of failure analysis. Our experimental validation focuses on predicting BEs and using these predictions to calculate the Top Event (TE) probability. The results demonstrate improved diagnostic accuracy and reliability, highlighting the potential of combining ML predictions with traditional FTA to identify root causes of failures in cloud computing environments and make the failure diagnostic more explainable.

## 1 INTRODUCTION


In the rapidly evolving domain of cloud computing, ensuring the reliability of systems has become a major concern among users (Mesbahi et al., 2018). As cloud services grow more complex, the potential for faults increases, making it crucial to employ sophisticated methods for fault detection and analysis (Ng'ang'a et al., 2023).


One traditional approach for understanding and mitigating system failures is Fault Tree Analysis (FTA). FTA utilizes a Fault Tree (FT), a graphical representation that describes the logical connections between various faults and their root causes through the use of logical gates. At the heart of the FT are Basic Events (BE), which are the fundamental fault conditions or failures that can occur within the system components. These BEs are interconnected through logical gates (such as AND, OR, NOT gates) that define how combinations of these BEs can lead to higher-level faults or system failures, ultimately leading to the Top Event (TE) or system failure. FTA is inherently deductive, starting with a system failure or TE and tracing back through the network of faults to identify root causes. This structured approach allows

for a comprehensive analysis of the pathways leading to system failures, emphasizing how combinations of component failures or specific environmental conditions can converge to trigger a system fault. By methodically breaking down the fault process from the TE to the BEs via logical gates, FTA provides a clear and detailed map of potential fault pathways, thereby facilitating targeted interventions to increase system reliability and prevent failures (Mani and Mahendran, 2017).

Simultaneously, the field of Machine Learning (ML) has shown great promise in enhancing the capabilities of fault detection and prediction in cloud computing environments (Yang and Kim, 2022). ML, particularly through its subfield of Deep Learning (DL), offers powerful tools for identifying patterns and anomalies in data that may indicate impending failures. However, many ML techniques, especially those involving DL, suffer from a lack of transparency. When these models predict a TE or system failure, they often do not provide insight into the underlying causes or the logical pathway leading to that prediction. This "black box" nature of ML especially DL models poses a significant challenge in fault analysis, where understanding the root causes is crucial for effective mitigation and prevention (Hoffmann and Reich, 2023).

Cloud computing infrastructures availability rely

<sup>a</sup>  <https://orcid.org/0000-0002-9061-5417>

<sup>b</sup>  <https://orcid.org/0000-0001-9831-2181>

on many components, like software, hardware, Cloud Management System (CMS), security, environmental, and human operation, etc. If something goes wrong the Root Cause Analysis (RCA) is often complex. To overcome these challenges, our research proposes an innovative integration of ML and FTA to enhance fault detection and analysis in cloud computing systems. This approach aims to combine the predictive power of ML with the systematic analysis capabilities of FTA, offering a pathway to not only predict system failures more accurately but also to provide insights into their underlying causes. Through this work, we try to bridge the gap between advanced computational models and interpretable fault analysis. The rest of our paper is structured as follows. Section 2 delves into the background, providing a comprehensive overview of FTA, the role of ML in fault detection, and the emerging significance of eXplainable Artificial Intelligence (XAI). This section also introduces the concept of surrogate models as a bridge between complex ML models and interpretable analysis. In section 3, we present our theoretical framework proposed in this work, describing how ML can be combined with FTs. This section lays the groundwork for integrating ML with FTA to achieve a transparent and interpretable fault detection system. In section 4, we conduct an experimental validation, where we test the approach of using ML for BE predictions and calculating the TE, demonstrating the practical application of our theoretical framework. In section 5 we present our results and discuss the benefits and challenges of our proposed theoretical frameworks. Finally, section 6 concludes our paper, summarizing key findings, and future research directions.

## 2 BACKGROUND

### 2.1 Fault Tree Analysis (FTA)

In cloud computing, the reliability of systems and the minimization of failures are crucial. FTA is an essential tool for systematically analyzing the factors contributing to system failures. A FT visually represents the logical relationships between various failure events, categorized into Intermediate Events (IEs) and BEs, which lead to a top-level failure, known as the TE. IEs represent combined underlying causes, while BEs denote fundamental root causes or failure modes. FTs employ logical gates like AND and OR to demonstrate how different events interact, influencing the occurrence of the top-level failure (Fazlollahabari and Niaki, 2018). Figure 1 illustrates typical examples of event symbols used in the FT structure. The

events in the FT are linked using gate symbols. Common gates are shown in figure 2 (Nieuwhof, 1975). Figure 3 represents an abstract FT that consists of these symbols as an example. Having the probabilities for the BEs, we can compute the TE. Let's break the formulas to calculate the probability for the TE down to use the probabilities for the BEs for that task. (Xie et al., 2021)

Both IEs are connected by an AND-gate. We can calculate them with:

$$P_{TE} = P_{IE1} \wedge P_{IE2} \tag{1}$$

The IEs can be calculated with the following formulas:

$$P_{IE1} = (P_{BE1} \vee P_{BE2}) \tag{2}$$

$$P_{IE2} = (P_{BE3} \vee P_{BE4}) \tag{3}$$

Now, let's use the probabilities for the BEs to calculate the probability for the TE.

$$P_{TE} = (P_{BE1} \vee P_{BE2}) \wedge (P_{BE3} \vee P_{BE4}) \tag{4}$$




Symbol	Name	Meaning
	Top Event (TE)	The primary failure or system malfunction being analyzed. It's the event of interest that the fault tree aims to prevent
	Intermediate Event (IE)	Represents failures that result from one or more lower-level events but can also cause other failures. They are part of the pathways leading to the top event.
	Basic Event (BE)	Refers to the most fundamental causes of the top event that cannot be decomposed further within the fault tree. These are often component failures or human errors.

Figure 1: Event symbols.



Symbol	Name	Causal Relation
	AND	Output event occurs if all input events occur.
	OR	Output event occurs if at least one of the input events occur.

Figure 2: Gate symbols.

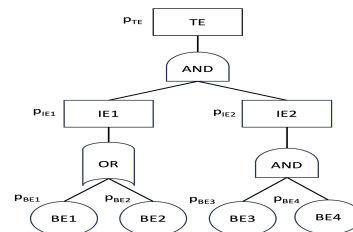


Figure 3: Example of a fault tree.

## 2.2 Artificial Intelligence (AI) and Explainable Artificial Intelligence (XAI)

The integration of Artificial Intelligence (AI), including ML and DL, has significantly advanced fault detection by analyzing complex data patterns to indicate potential system issues. However, the opacity of DL models, often described as "black box" systems, poses a challenge in understanding, interpreting and trusting their predictions. This opacity has catalyzed a shift towards XAI, that aim to make the decision-making processes more understandable. It emphasizes the need for transparency and interpretability in AI systems. By explaining the relationships between input variables and the failure outcomes, it helps identify the underlying causes of failures. XAI aims to bridge the gap between AI's complex algorithms and user comprehensibility, ensuring that the rationale behind AI decisions is transparent, fostering trust and wider acceptance in AI-driven solutions (Hoffmann and Reich, 2023).

Surrogate models, as a method within XAI, serve as an interpretable approximations of complex AI systems. These models, also known as response surfaces or meta-models, are utilized to simplify the relationships between input and output data. This simplification is particularly valuable when the actual connections are unknown or too complex to compute efficiently. By applying surrogate models, XAI aims to make AI's decision-making processes more transparent and understandable, enhancing user trust and facilitating more informed decision-making in critical applications (Williams and Cremaschi, 2019).

## 3 THEORETICAL FRAMEWORK

### 3.1 The Role of Fault Trees in Surrogate Model-Based Fault Analysis

In section 2 we explained that surrogate models act as interpretable approximations of complex models, providing insights into how inputs affect outputs. Similarly, FTs systematically map the relationships between BEs and the TE, offering a clear view of causal pathways. Our approach leverages ML models to predict BEs within the FT framework. By predicting these BEs, we gain insight into the specific events or conditions that directly contribute to the system failure. Subsequently, FTs are employed to compute the likelihood of the TE based on the occurrence of these predicted BEs. Moreover, ML tech-

niques can aid in the selection or generation of FTs of complex systems. This integration of ML with FT enhances the traceability and comprehension of failure occurrences, facilitating the identification of root causes. Thus, our approach not only enhances the transparency of failure detection but also enables a deeper understanding of failure mechanisms within complex systems.

### 3.2 Combining Fault Trees with Machine Learning

In this section we describe the different combination methods in more detail.

#### (A) Machine Learning for the Fault Tree Selection

In the field of cloud computing, navigating through multiple failure scenarios efficiently is pivotal due to the complex interaction of system components and external variables. This complexity makes it necessary to use an automated method to identify suitable FTs in a collection of FTs or from a huge FT (see Figure 4). Instead of relying solely on manual expertise or predefined rules, ML algorithms analyze observed symptoms or failure modes to match them with the most appropriate FT. This predictive capability significantly enhances the fault diagnosis process by narrowing down the search space and pinpointing potential root causes. Importantly, by automating this selection process, we reduce the influence of subjective biases, ensuring more objective and consistent fault diagnosis. Furthermore, by selecting the best-suited FT, our approach indirectly leverages it as a surrogate model to approximate the underlying failure mechanisms. This surrogate model aids in making complex diagnostics more manageable, providing insights into the causal relationships between various system events and failures. However, this strategy requires the availability of multiple expert FTs, underscoring the need for a rich repository of FTs to cover the spectrum of potential failures in cloud computing environments.

#### (B) Machine Learning for the Fault Tree Generation

In this method, we use observational or historical data to automate the generation of FTs that encapsulate the system's failure modes, thereby serving as a surrogate model (see Figure 5).

By leveraging ML techniques, we can derive insights from the data to construct FTs that accurately represent the complex relationships between system

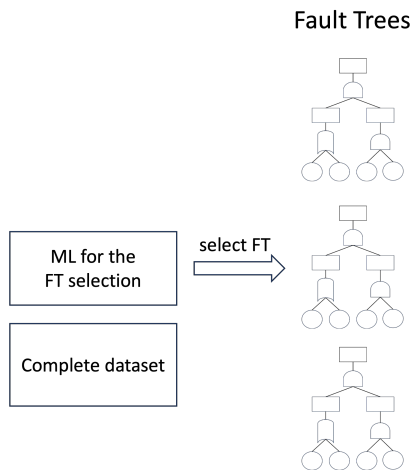


Figure 4: Using ML for the FT selection.

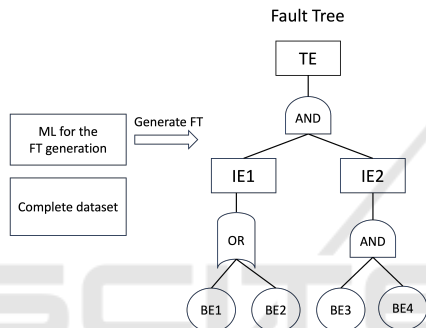


Figure 5: Using ML for the FT generation.

components and failure events. The use of observational or historical data enables us to capture real-world scenarios and patterns, facilitating the creation of comprehensive FTs. However, it's crucial to ensure that these generated FTs strike a balance between interpretability and relevance. This often involves refining the FTs by simplifying or pruning excessive details to enhance clarity without compromising the representation of critical failure pathways. Moreover, generating an effective FT requires the integration of expert knowledge to ensure alignment with the system's failure modes. This fusion of ML-driven data analysis with expert insights enhances the accuracy and relevance of the generated FTs, enabling them to serve as valuable tools for fault diagnosis and system understanding. However, generating a FT with expert knowledge and ensuring it accurately represents the system's failure modes, can be difficult. Despite these challenges, the automated generation of FTs through ML offers a powerful means of capturing and understanding the underlying mechanisms of system failures, ultimately facilitating more effective analysis and decision-making in fault diagnosis and system maintenance.

### (C) Machine Learning for the Fault Tree Generation and Selection

This approach merges the generation and selection of FTs through ML (see Figure 6). ML algorithms are employed to generate FTs based on observational or historical data, and then to select the most fitting FT for a given situation. This strategy aims to enhance the efficiency of diagnosing system failures by leveraging ML's capability to analyze complex data and identify significant patterns, thereby providing an analysis tool for different failure scenarios.

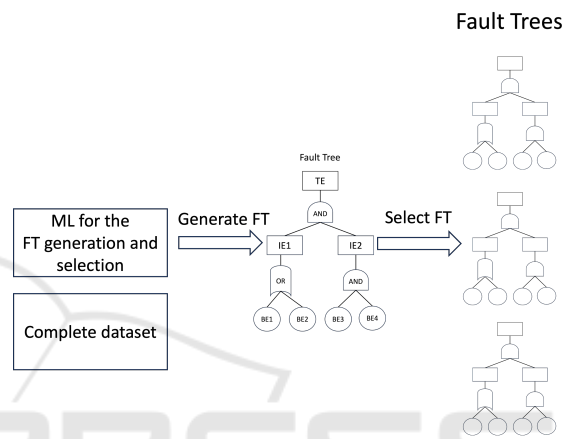


Figure 6: Using ML for the FT generation and selection.

### (D) Machine Learning for the Basic Event Prediction

This approach utilizes ML models to predict BEs within FTs, translating these predictions into probabilities to determine the TE's likelihood (see Figure 7). By predicting BEs of the superior events like the TE, allows the identification of root causes behind failure occurrences. Furthermore, the deductive nature of FTs allows to determine the TE and thus acts as a surrogate model, thereby boosting the explainability of TE predictions. This mechanism not only enhances the explainability of TE predictions but also provides insights into the causal relationships between individual events and system failures. Furthermore, this approach leverages the adaptability of ML models to continually refine prediction accuracy through iterative data learning. By incorporating new data and insights, the ML models can dynamically adjust their predictions, improving the accuracy and reliability of failure predictions over time. In essence, this method explains failure modes and their connections within complex systems. By combining the interpretability of FTs with the predictive power of ML, our approach offers understanding and addressing system failures in diverse environments.

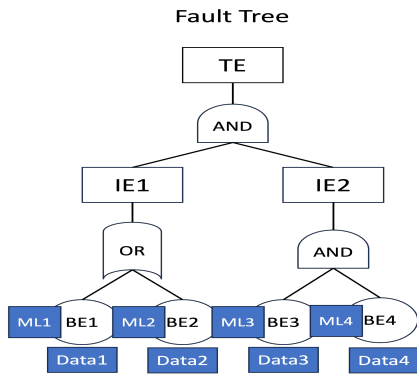


Figure 7: Using ML for the BEs prediction.

### 3.3 Overview of all Combinations

Table 1 provides an overview of the described combination cases. It describes, in what way the FT acts as a surrogate model.

Table 1: FT acting as a surrogate model in different combination cases.

Combination	FT Act as Surrogate Model
(A)	By selecting a FT, the system indirectly uses it to approximate the underlying failure mechanism, making complex diagnostics more manageable. This conceptualization of the FT as a surrogate model aids in simplifying fault analysis and identifying root causes effectively.
(B)	The generated FTs act as surrogate model by modeling the system’s complex failure mechanisms through a structured and simplified representation.
(C)	Integrates both the generation and selection of FTs.
(D)	The FT acts as a surrogate model by providing a simplified, yet effective, representation of the system’s failure mechanisms. The FT allows the estimation of TEs based on BE probabilities, which can be seen as approximating the overall system’s failure behavior through a more manageable and interpretable framework.

## 4 EXPERIMENT

### 4.1 Fault Tree Selection

In the paper (Mesbahi et al., 2018), diverse failure classifications within cloud computing systems are detailed, including software, hardware, CMS, security, environmental, and human operation failures, along with their respective modes. Based on this comprehensive classification, we constructed a FT with "Cloud System Failure" as the TE, categorized the failure classifications as IEs, and detailed their modes as BEs, as illustrated in Figure 8. Drawing from our theoretical framework in Section 3, our validation focuses on the "Hardware Failure" class. We simplified the overarching FT by isolating the "Hardware Failure" branch, yielding a focused sub-tree that is used for the proof of concept of our approach ML for the BE prediction (see section 3.2). You see the focused FT in Figure 9. The hardware failures occurs, if a hardware component (hard drive in this case) or network indicates a failure. While network failures can occur from various sources, not just hardware issues, for our experiment, we proceed with a specific assumption. This focus allows us to streamline our analysis within the context of our FTA, concentrating on hardware-related aspects to provide clarity and specificity to our investigation.

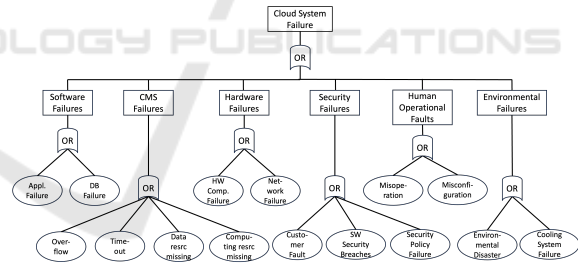


Figure 8: Cloud System Failure FT based on the description in (Mesbahi et al., 2018).

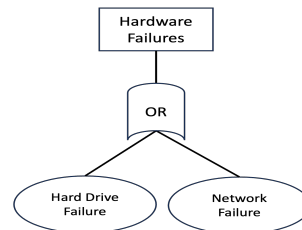


Figure 9: FT focusing on the hardware failure.

## 4.2 Dataset Description

### 4.2.1 SOFI Dataset

The SOFI (Symptom-Fault relationship for IP-Network) dataset contains information about an extensive enterprise network's performance, indicating well-known faults across various times and days, totaling approximately 649 hours of monitoring. Notably, 10 hours of this dataset capture periods when faults were intentionally induced to study their impact. The dataset includes 34 attributes covering performance metrics and fault indicators, classifies network status into faulty (F) or healthy (NE), and comprises 12,971 instances, offering a rich resource for analyzing network fault dynamics and developing fault detection models (Vargas-Arcila et al., 2021).

### 4.2.2 SMART Dataset

The dataset encompasses S.M.A.R.T. attributes from four distinct hard drives within the BackBlaze Data Center, detailing aspects like model, serial number, date, and capacity, all preprocessed for analysis. The dataset specifically contains records of failed Seagate hard drive S.M.A.R.T. information, with data on 56 attributes across 128,818 failure instances and 1,031,502 instances indicating normal operation, providing a valuable dataset for predicting hard drive failures. (Backblaze, 2023)

## 4.3 Merging Datasets

To integrate the SOFI network dataset with the S.M.A.R.T. hard drive dataset from BackBlaze, we adopted an approach to merge the dataset, aimed at analyzing the interplay between network and hard drive health. This process involved horizontally merging features of operational (good) hard drives and networks, appending indicators (class\_hd=0, class\_nw=0, class\_hw=0) to imply the absence of failures. Conversely, combinations of operational and faulty states between hard drives and networks were similarly merged, with appended classifications to reflect the presence or absence of failures in each domain, thereby enabling a comprehensive analysis of hardware health in relation to network and hard drive performance. The merged dataset contains 25942 records with 90 attributes.

## 4.4 Modeling

In our experiment, we compared two modeling approaches. In the first approach, we used the merged

dataset with a DL model to predict, if a hardware failure exist. In the second approach, we tried the proposed approach to use DL models to predict the BEs and then determine the TE. For both approaches, we used the same model architecture. The architecture is shown in Table 2. We created the DL model using TensorFlow and Keras. For the architecture, we used four sequential Dense layers. We used Rectified Linear Unit (ReLU) as activation function for the hidden layers, while Sigmoid for the classification layer to constrain output between zero and one. Additionally, we adopted a k-fold cross-validation strategy with 10 splits to ensure the robustness and generalizability of our model across different subsets of the data. This methodological choice aims to mitigate overfitting and assess the model's performance more accurately. The hyperparameters used to build the model were: (Hoffmann et al., 2022)

**optimizer:** Adam with a learning rate of 0.001

**loss:** 'binary\_crossentropy'

**epochs:** 30

**batch\_size:** 32

Table 2: Architecture of the DL Model.

Layer	Units	Activation Function
Dense1	128	ReLu
Dense2	64	ReLu
Dense3	32	ReLu
Dense4	1	Sigmoid

### 4.4.1 Approach 1 - Predicting the Top Event

In this common approach, we utilize the merged dataset, comprising 90 attributes, to directly predict the target variable 'class\_hw', which indicates the presence of a hardware failure. This prediction is made by the DL model described in Table 2. This method uses a comprehensive dataset to predict the hardware failure risk using a singular predictive model.

### 4.4.2 Approach 2 - Predicting the Basic Events

In this new approach described in our theoretical framework (see section 3.2) we utilize two DL models with the architecture described in Table 2. The first model uses attributes of the hard drive to predict, whether a hard drive failure exists (class\_hd). The other model uses the other attributes to predict, whether a network failure exists (class\_nw). The confidence values of both predictions are used to calculate the confidence value of the TE (hardware failure). We treat the confidence values as probabilities of a FT

and calculate the probabilities of both BEs:

$$P_{hw} = P_{hd} \vee P_{nw} \quad (5)$$

Both events are independent from each other. Thus, we can calculate it with: (Kaptein and van den Heuvel, 2022)

$$P_{hw} = P_{hd} + P_{nw} - (P_{hd} \times P_{nw}) \quad (6)$$

Classifying the hardware failure using this approach makes the prediction more explainable, since the failures, that lead to this occurrence, are known. After predicting the BEs (hard drive and network failure), the FT acts as a surrogate model.

## 5 RESULTS AND DISCUSSION

The results presented in this section represent the mean values obtained after executing the algorithms ten times. This approach was chosen to ensure the reliability and stability of our findings, aiming to account for variability in performance across different runs. By averaging the outcomes, we tried to provide a more accurate and robust assessment of the modeling approaches. Table 3 compares the results of the different approaches.

The results indicate that both modeling approaches yield excellent outcomes, with the proposed method (predicting BE and calculating the TE) slightly outperforming the traditional approach across all metrics: accuracy, precision, recall, F1-score, and Area Under the ROC (Receiver Operating Curve) Curve (AUC-ROC). Crucially, the proposed approach offers additional value by identifying the root causes of the TE failure, enhancing the interpretability of the results. This contrasts with the common approach, which predicts the occurrence of the TE without indicating the underlying reasons for its occurrence.

Table 3: Results of our Experiments.

Metric	TE Prediction	BEs Prediction
Accuracy	99.1 %	99.4 %
Precision	99.7 %	99.8 %
Recall	98.6 %	99.1 %
F1-Score	99.2 %	99.5 %
AUC-ROC	99.9 %	99.6 %

In this study, we explored four methods to integrate ML with FTs, but our experimental validation focused solely on the technique of using ML to predict BEs. The potential approaches involving ML for selecting, generating, or both selecting and generating FTs were not explored in this work. Instead, we

concentrated on predicting the BEs within an existing or readily available FT, demonstrating the practical application and benefits of this specific approach in enhancing fault diagnosis.

Although we validate only one approach, we want to discuss the challenges and benefits of all approaches described in section 3. The first approach, utilizing ML to select the most appropriate FT, presents a strategic advantage in narrowing down the search space for RCA. This way, the FT approximates the underlying failure mechanism. Acting as a surrogate model, it enhances diagnostic efficiency and reduces the reliance on computational resources. This method, however, faces challenges in managing the complexity inherent in FTs, especially as system dynamics evolve, requiring continuous updates and adjustments.

The second strategy, employing ML for the automated generation of FTs, marks a significant shift towards reducing dependency on expert knowledge for FT construction. This approach not only streamlines the fault diagnosis process, but also opens ways for uncovering hidden patterns and relationships within system's complex failure mechanisms by modeling it using FTs, offering a new perspective on system improvements. Since the FT models complex failure mechanisms, it can be viewed as a surrogate model. Despite these benefits, the risk for generating complex or redundant FTs poses a significant challenge, emphasizing the need for sophisticated post-processing techniques to ensure the usability and interpretability of the generated trees. Additionally, generating a FT with expert knowledge and ensuring it accurately represents the system's failure modes can be difficult.

Combining the generation and selection of FTs through ML, our third approach attempts to harness the strengths of both mentioned strategies. This integrated method promises a comprehensive solution to fault diagnosis, but it introduces complexity in effectively merging these processes, particularly in verifying the appropriateness of the selected or generated FTs.

Our fourth and final approach focuses on employing ML to predict BEs within the FT framework, significantly enhancing the fault diagnosis's reliability and interpretability, since the FT allows the estimation of the TE based on BE probabilities and thus act as a surrogate model. This method allows the identification of root causes and offers the understandability of the TE's occurrence, thereby increasing the transparency of the entire process. However, it's important to note that while this approach brings explainability to the occurrence of the TE, the occurrence of the BEs

themselves remains opaque. The "black box" nature of DL models used for predicting these events limits our ability to fully understand and interpret the occurrence of the BEs.

In future work, our research will explore the unvalidated approaches of using ML for selecting, generating, or both selecting and generating FTs. We will investigate methodologies for employing ML algorithms to automate the selection of appropriate FTs based on observed symptoms or failure modes. This will involve developing algorithms that navigate through multiple failure scenarios to identify the most suitable FTs for RCA. Furthermore we will investigate how ML can be utilized to automate the generation of FTs based on observational or historical data. This involves developing algorithms that construct FTs that accurately represent the complex failure mechanisms within cloud computing systems, while also ensuring interpretability and relevance for effective fault diagnosis. By pursuing these paths, we aim to enhance fault diagnosis by fully leveraging the integration of ML with FTs. Additionally, we will explore the implementation of our approach in real-world settings to evaluate its applicability and robustness across various cloud computing environments. Through these efforts, we try to unlock advanced capabilities for more precise analysis and understanding of system failures.

## 6 CONCLUSION

Our investigation into integrating ML with FTA presents a significant advancement in fault detection methodologies for cloud computing systems. By concentrating on the prediction of BEs and the subsequent calculation of TE probability, we not only enhance the precision of fault diagnosis but also increase the system's interpretability and transparency. Although our experimental validation focused on this particular approach, we discussed the theoretical framework and potential benefits of using ML for selecting and generating FTs. Future work will explore these unvalidated approaches to further refine and expand our understanding of integrating ML with FTA, aiming to develop more robust and intuitive fault diagnosis tools for complex computing environments.

## FUNDING

This research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), under grant DFG -GZ: RE 2881/6-1

and the French Agence Nationale de la Recherche (ANR), under grant ANR-22-CE92-0007.

## REFERENCES

- Backblaze (2023). Harddrive cleaned smart dataset. Accessed: 2024-02-15.
- Fazlollahtabar, H. and Niaki, S. (2018). Fault tree analysis for reliability evaluation of an advanced complex manufacturing system. *Journal of Advanced Manufacturing Systems*, 17:107–118.
- Hoffmann, R. and Reich, C. (2023). A systematic literature review on artificial intelligence and explainable artificial intelligence for visual quality assurance in manufacturing. *Electronics*, 12(22).
- Hoffmann, R., Reich, C., and Skerl, K. (2022). Evaluating different combination methods to analyse ultrasound and shear wave elastography images automatically through discriminative convolutional neural network in breast cancer imaging. *International Journal of Computer Assisted Radiology and Surgery*, 17(12):2231–2237.
- Kaptein, M. and van den Heuvel, E. (2022). *Probability Theory*, pages 81–102. Springer International Publishing, Cham.
- Mani, D. and Mahendran, A. (2017). An approach to evaluate the availability of system in cloud computing using fault tree technique. *International Journal of Intelligent Engineering and Systems*, 10:245–255.
- Mesbahi, M. R., Rahmani, A. M., and Hosseinzadeh, M. (2018). Reliability and high availability in cloud computing environments: a reference roadmap. *Human-centric Computing and Information Sciences*, 8(1):20.
- Ng'ang'a, D. N., Cheruiyot, W., and Njagi, D. (2023). A machine learning framework for predicting failures in cloud data centers -a case of google cluster -azure clouds and alibaba clouds. Accessed: 2024-02-17.
- Nieuwhof, G. (1975). An introduction to fault tree analysis with emphasis on failure rate evaluation. *Microelectronics Reliability*, 14(2):105–119.
- Vargas-Arcila, A. M., Corrales, J. C., Sanchis, A., and Rendón, A. (2021). Dataset of symptom-fault causal relationships for an ip-based network. Accessed: 2024-02-15.
- Williams, B. and Cremaschi, S. (2019). Surrogate model selection for design space approximation and surrogate-based optimization. In Muñoz, S. G., Laird, C. D., and Realf, M. J., editors, *Proceedings of the 9th International Conference on Foundations of Computer-Aided Process Design*, volume 47 of *Computer Aided Chemical Engineering*, pages 353–358. Elsevier.
- Xie, X., Wang, Y., Hu, K., and Du, J. (2021). Quantitative analysis of fault diagnosis based on fault tree reasoning. In *2021 3rd International Conference on Applied Machine Learning (ICAML)*, pages 7–10.
- Yang, H. and Kim, Y. (2022). Design and implementation of machine learning-based fault prediction system in cloud infrastructure. *Electronics*, 11(22).